HALMA 1.1

Hanover Logic Minimization Algorithms

Handbuch für Benutzer

Version August '86

Axel Kemper

Universität Hannover

Institut für Theoretische Nachrichtentechnik
und Informationsverarbeitung

Callinstrasse 32/1305, Ruf: (0511) 762-5305

0 **Einführung** 0-5
0.1 Entwurf von Schaltnetzen 0-5
0.2 HALMA 0-8
0.3 Aufbau und Benutzung des Handbuchs 0-8

1 **Die Benutzung von HALMA** 1-1
1.1 Bevor Sie beginnen 1-1
1.2 Die Dateien auf dem HALMA-Band 1-1
1.3 Start von HALMA 1-2
1.4 Die Installation 1-2
1.4.1 Lesen des Magnetbandes 1-3
1.4.2 DCL-Symbolvereinbarungen 1-3
1.5 Das HALMA-Menü 1-4
1.5.1 Hilfe! 1-5
1.5.2 Programmteile auswählen 1-6
1.5.3 Starten eines Programmteils 1-6
1.5.4 Editieren der Eingabe 1-6
1.5.5 Ansehen der Ausgabe 1-6
1.5.6 Der Projektname 1-7
1.5.7 Stapelverarbeitung 1-7
1.5.8 Ergebnisse ausdrucken 1-8
1.5.9 DCL-Kommandos zwischendurch 1-8
1.5.10 Ausgabe von Plots 1-8
1.5.11 HALMA beenden 1-9
1.5.12 Wenn ein Fehler eintritt 1-9

2 **HDL: Übersetzung von Schaltnetzspezifikationen** 2-1
2.1 Einführung 2-1
2.2 Die Spezifikationssprache 2-1
2.2.1 PROGRAM 2-2
2.2.2 DECLARE: Variable vereinbaren 2-2
2.2.3 CONSTRAINTS: Wertebereiche festlegen 2-3
2.2.4 DELAYS: Laufzeiten vorgeben 2-3
2.2.5 SPECIFICATION: die eigentliche Beschreibung 2-4
2.2.5.1 Wertzuweisungen 2-4
2.2.5.2 Arithmetische Operationen 2-4
2.2.5.3 Logische Operationen 2-5
2.2.5.4 IN und OUT 2-5

2.2.5.5 IF .. THEN .. ELSE .. ENDIF 2-6
2.2.5.6 CASE .. ENDCASE 2-6
2.3 Das Übersetzungsprotokoll 2-7
2.4 Editieren der Eingabe 2-7
2.5 Wenn ein Fehler eintritt 2-7

3 **HLM: Synthese von unvermaschten NAND2-Inverter-Netzen** 3-1
3.1 Einführung 3-1
3.2 Die Eingabe-Syntax im LOGE-Format 3-1
3.2.1 Deklaration der Schaltnetzparameter 3-2
3.2.2 Die Funktionstabelle 3-2
3.2.3 Formeleingabe 3-3
3.2.4 Optimierungskriterien 3-4
3.2.5 Heuristik-Parameter 3-5
3.2.6 Synthese-Arten 3-5
3.2.6.1 Zweistufige oder mehrstufige Synthese? 3-6
3.2.6.2 Bündel- oder Einzel-Optimierung? 3-6
3.2.7 Im Falle von Eingabefehlern 3-7
3.3 Bestandteile der Ausgabe 3-7
3.4 Die Syntax der erzeugten Netzliste(n) 3-8
3.5 Wenn HLM keine Lösung findet 3-8
3.6 Im Fehlerfall 3-8

4 **TREE: Optimierung durch lokale Transformationen** 4-1
4.1 Einführung 4-1
4.2 Drei Optimierungs-Modi 4-1
4.3 Gewichtung der Optimierungskriterien 4-2
4.4 Eingabe der Technologie-Parameter 4-2
4.5 Steuerung der Protokoll-Ausgabe 4-3
4.6 Im Fehlerfall 4-3

5 **PLT: Plot-Ausgabe von Schaltnetzen** 5-1
5.1 Einführung 5-1
5.2 Zwei Normen für Schaltzeichen 5-1
5.3 Druck- oder Plot-Ausgabe 5-1
5.4 Ausgabe auf Tektronix-Terminals 5-2
5.5 Ausgabe in HP-GL 5-2

5.6 Im Fehlerfall 5-3

6 Literatur zu HALMA 6-1

Anhänge:

A HDL-Syntax im Überblick A-1
B LOGE-Syntax mit Erweiterungen B-1
C Das HALMA Netzlisten-Format C-1
D Vordruck für Benutzer-"Feedback" D-1

0 Einführung

Das CAD-Programm HALMA wurde in den Jahren 1983 bis 1986 am Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung der Universität Hannover im Rahmen des E.I.S.-Projektes entwickelt. HALMA automatisiert den Entwurf von Schaltnetzen und gestattet dabei dem Benutzer, neben technologischen Parametern Optimierungsgewichte und Randbedingungen für die Verzögerungszeit vorzugeben.

Dieses Handbuch beschreibt die Benutzung von HALMA und ist für Leser gedacht, die schon eine gewisse Erfahrung im Umgang mit Rechnern haben. Insbesondere wird die Bedienung von VAX-Rechnern (Editieren, Starten von Programmen, BACKUP etc.) als bekannt vorausgesetzt.

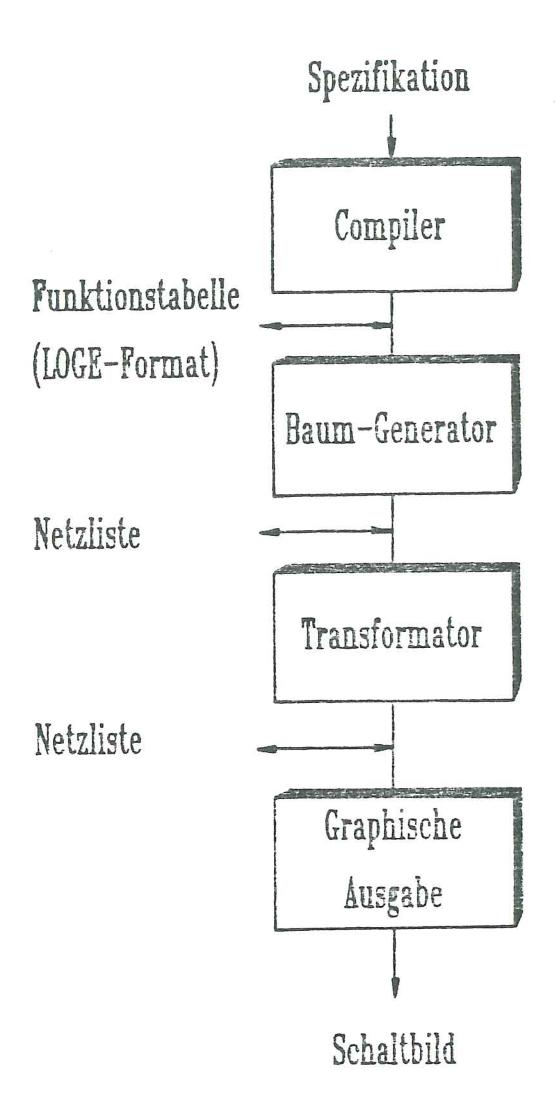
Sollten Ihnen Teile dieses Handbuches zu knapp erscheinen, so muß auf die in Kapitel 6 aufgeführte Literatur verwiesen werden. HALMA entstand aus Arbeiten von Th.Frank, A.Kemper, B.Krönke und Th.M.Sarfert. Die jeweiligen Original-Arbeiten enthalten detaillierte Beschreibungen der Bestandteile von HALMA.

0.1 Entwurf von Schaltnetzen

HALMA führt den Schaltnetzentwurf in vier Schritten durch:

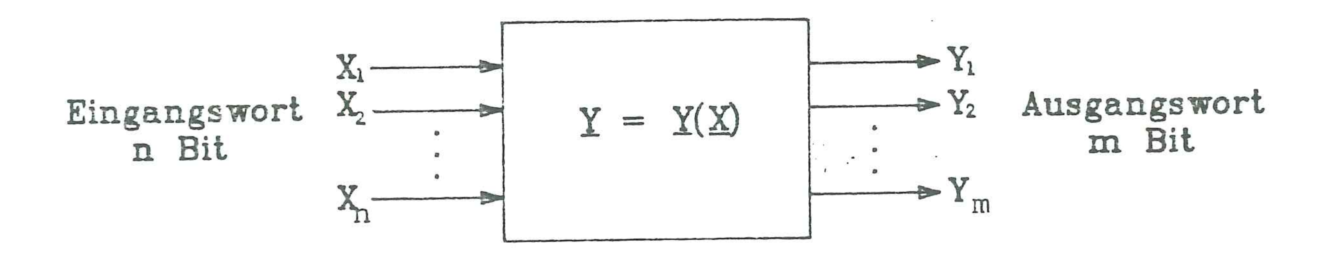
1. Übersetzung der Schaltnetzspezifikation in eine Funktionstabelle.
2. Umsetzung der Funktionstabelle in unvermaschte Schaltnetze aus NAND-Gattern mit je zwei Eingängen und Invertern.
3. Optimierung der NAND2-Inverter-Schaltnetze durch lokale Transformationen und durch Zusammenfassen äquivalenter Unterschaltnetze.
4. Graphische Ausgabe der erzeugten Schaltnetze als Druck- oder Plot-Graphik.

Aufbau von HALMA



Da ein Schaltnetz ein **n**-Bit Eingangs-Wort gedächtnislos in ein **m**-Bit Ausgangs-Wort umcodiert, kann man jedes Schaltnetz durch Angabe arithmetischer oder logischer Verknüpfungen eindeutig spezifizieren. HALMA enthält zu diesem Zweck einen Compiler, der in Schritt 1. in einer Beschreibungssprache abgefaßte Spezifikationen in eine Art Funktionstabelle umsetzt. Die Funktionstabellen haben das sogenannte LOGE-Format (LOGE ist ein CAD-Programm von der Uni-

versität Karlsruhe).



Die Funktionstabelle wird im zweiten Schritt auf verschiedene Arten, unter denen der Benutzer wählen kann, zusammengefaßt bzw. minimiert und in ein mehrstufiges Schaltnetz umgesetzt, daß nur aus Invertern und NAND-Gattern mit je zwei Eingängen besteht. Solche Schaltnetze sind in der CMOS-Technologie recht verzögerungsarm und lassen sich mit vertretbarem Aufwand entwerfen. Der Benutzer kann vorgeben, wieviele Stufen das Schaltnetz maximal haben soll. Außerdem kann er die verwendeten Such-Heuristiken durch Setzen von Parametern beeinflussen.

Die NAND2-Inverter-Schaltnetze werden im dritten Schritt weiter optimiert. Es wird versucht Teilschaltnetze zu finden, die sich durch günstigere Gatterkonstellationen ersetzen lassen. Durch Parametrisierung einer Gütefunktion kann der Benutzer bestimmen, welche Schaltungseigenschaften (Fläche, Delay, Gatterzahl, Zahl der Transistoren, Zahl der Verbindungen) bevorzugt optimiert wird.

Um die Entwurfsergebnisse überprüfen zu können, enthält HALMA eine graphische Ausgabe. Die ermittelten Schaltpläne können als Drucklisten oder Plots ausgegeben werden. Zur Zeit enthält HALMA Plot-Treiber für Tektronix-Terminals, für HP-GL (Hewlett-Packard Graphics Language) und für die im Institut für Theoretische Nachrichtentechnik verwendete Plot-Schnittstelle. Weitere Treiber lassen sich leicht hinzufügen, da nur mit Low-Level-Primitiven geplottet wird.

Die vier Teilprogramme von HALMA können auch einzeln verwendet werden, wenn ihre Eingabedaten durch andere CAD-Programme oder von Hand erzeugt wurden. Die Schnittstellenformate für Netzlisten und Funktionstabellen werden in den entsprechenden Abschnitten und im Anhang näher beschrieben.

0.2 HALMA

HALMA wurde auf einer VAX11/780 implementiert und bisher außerdem auf einer MicroVAX II und einer VAX11/785 erprobt. Der Betrieb auf Rechnern, die nicht der VAX-Familie angehören, wird momentan nicht unterstützt. An der Implementierung einer abgemagerten Version von HALMA auf dem IBM PC wird gearbeitet.

HALMA enthält neben den in 0.1 aufgeführten vier Programmen diverse DCL-Prozeduren. Die wichtigste von ihnen ist HALMA.COM. HALMA.COM dient als Benutzeroberfläche und steuert die Aufrufe der Programme und die Verwaltung der Dateien.

Insgesamt besteht HALMA aus etwa 60000 Zeilen Source-Code in den Programmiersprachen C, FORTRAN 77, PASCAL und DCL. Bisher unentdeckte Programmierfehler sind daher nicht auszuschließen.

Sollten also bei der Benutzung von HALMA Fehler auftreten, die nicht auf fehlerhafte Benutzereingaben zurückzuführen sind, so wäre allen HALMA-Benutzern sehr damit gedient, wenn dies den Entwicklern von HALMA per Fehler-Report mitgeteilt würde. Die Kontaktadresse und einen entsprechenden Vordruck enthält der Anhang dieses Handbuches.

0.3 Aufbau und Benutzung des Handbuchs

Die Benutzer-Schnittstelle von HALMA wurde möglichst selbstklärend gehalten. Daher gibt das vorliegende Handbuch einen bewußt kurzen Überblick über die Benutzung von HALMA.

Das Handbuch ist in sieben Kapitel und vier Anhänge gegliedert:

Kapitel 0 ist die Einführung, die Sie gerade lesen.

Kapitel 1 behandelt die Benutzeroberfläche von HALMA und beschreibt, wie man HALMA installiert, startet, bedient und beendet.

Kapitel 2 beschreibt die Spezifikation von Schaltnetzen in einer Sprache, die vom HALMA-Teilprogramm HDL in eine Funktionstabelle umgesetzt werden kann.

Kapitel 3 ist der Synthese von unvermaschten NAND2-Inverter-Schaltnetzen gewidmet und erläutert die möglichen Benutzer-Eingriffe zur Beeinflussung des Synthese-Vorgangs.

Kapitel 4 beschreibt den dritten Block von HALMA zur Optimierung durch lokale Transformationen. In diesem Kapitel wird auch behandelt, wie technologische Daten der verwendeten Gattertypen für HALMA aufbereitet werden müssen.

Kapitel 5 schließt die Beschreibung der Teile von HALMA mit der Graphik-Ausgabe ab.

Kapitel 6 enthält eine Liste aller bisher erschienenen Arbeiten aus dem HALMA-Projekt.

Die **Anhänge** richten sich mehr an erfahrene HALMA-Benutzer(innen), denen Details entfallen sind. Das Formular in Anhang D sollte für Fehlermeldungen verwendet werden.

1 Die Benutzung von HALMA

1.1 Bevor Sie beginnen

Damit Sie HALMA benutzen können, muß HALMA sowohl auf Ihrem Rechner als auch in Ihrer DCL-Symboltabelle ordnungsgemäß installiert und vereinbart sein.

Wichtig: Sorgen Sie dafür, daß eine Sicherheitskopie des HALMA-Magnetbands an einem sicheren Ort verwahrt wird!

1.2 Die Dateien auf dem HALMA-Band

Die wichtigsten Dateien auf dem HALMA-Magnetband sind:

HALMA.COM	DCL-Steuerprozedur
HLMLOGIN.COM	definiert die DCL-Symbole
AUTOINSTALL.COM	korrigiert HLMLOGIN.COM
HDLRUN.EXE	Spezifikations-Compiler
HALMA.EXE	Synthese von NAND2-Inverter-Schaltnetzen
TREE.EXE	Lokaler Transformierer
SETUP.TREE	Parameter für TREE
GDVM.DAT	Technologiedaten für TREE
HALMAPLOT.EXE	graphische Ausgabe ("Halbkäse")
HALMAPLOTNEU.EXE	("Kästchen")
HALMADRUCK.EXE	(Druckliste)
HILONET.COM + .EXE	Netzlistenkonverter ==> HILO3
TEKPLOT.COM + .EXE	Plot-Treiber für Tektronix
HPPLOT.COM + .EXE	HP-GL
HALMADF.DAT	City-Block-Daten für HLM
MANUAL.DOC	dieses Handbuch
BUGS.DOC	bekannte Mängel von HALMA

Die Standard-Extensions für Dateinamen sind unter Punkt 1.5.6 aufgeführt.

1.3 Start von HALMA

Benutzer von HALMA sollten in ihrem LOGIN-File eine Anweisung der Form:

HALMA:==@PROG:[KEMDIR.HALMA.COM]HLMLOGIN.COM path

stehen haben. Durch Eingabe von HALMA wird dann zunächst HLMLOGIN.COM aufgerufen um die DCL-Symbole für HALMA zu vereinbaren. HLMLOGIN.COM definiert das Symbol HALMA so um, daß beim nächstenmal HALMA.COM direkt aufgerufen wird. Nach den Symboldefinitionen ruft HLMLOGIN.COM die Steuerprozedur HALMA.COM auf und startet damit HALMA auf der durch den Parameter "path" angegebenen Directory. Der Benutzer wird nach "project" gefragt, damit HALMA alle Dateinamen einheitlich mit dem Projektnamen versehen kann. Beim nächsten Aufruf von HALMA wird automatisch das zuvor verwendete Projekt übernommen, wenn kein neuer Name als erster Parameter übergeben wurde.

Beispiele:

@[...HALMA.COM]HLMLOGIN disk:[user.cad]

HALMA fulladder

1.4 Die Installation

HALMA geht von folgender Directory-Struktur aus:

[...HALMA]	HALMA Haupt-Directory
[...HALMA.COM]	DCL-Programme und Editor-Startups
[...HALMA.EXE]	exekutierbare Images
[...HALMA.DAT]	Hilfsdateien (Technologie etc.)

Wenn Sie die Instruktionen im folgenden Abschnitt 1.4.1 genau befolgen, wird diese Struktur bei Verwendung eines original HALMA-Magnetbandes automatisch eingerichtet und sollte nur verändert werden, wenn sehr gute Gründe dafür sprechen!

1.4.1 Lesen des Magnetbandes

Das HALMA-Magnetband wird mit folgenden DCL-Befehlen gelesen, nachdem es (ohne Schreibring!) in eine Bandmaschine mit der richtigen Aufzeichnungsdichte eingelegt wurde (1600 oder 6250 BpI je nach Band-Aufschrift):

```
ALLOCATE name
MOUNT/FOREIGN name
BACKUP/LOG/SELECT=[KEMDIR.HALMA...] -
   name:HALMA.BKP disk:[dir.HALMA...]
DISMOUNT/UNLOAD name
DEALLOCATE name
```

Dabei ist:

name - der Name der Bandstation (z.B. MTA0:)
disk - die Platte, auf der HALMA installiert werden soll (z.B. DRB1:)
dir - die Directory, unter der HALMA auf "disk" installiert werden soll (z.B. SYS$CAD)

1.4.2 DCL-Symbolvereinbarungen

Damit HALMA gestartet werden kann, muß die Prozedur HLMLOGIN.COM ein CONCEALED ASSIGN durchführen, das die Platte HLM$DISK definiert. Sämtliche Dateizugriffe von HALMA beziehen sich auf diese Platte, so daß nur eine Zeile in HLMLOGIN.COM auf der Directory [...HALMA.COM] geändert werden muß, um HALMA auf einer anderen Platte zu installieren. Wenn Sie sich auf der Directory [..HALMA.COM] befinden und mit

@AUTOINSTALL

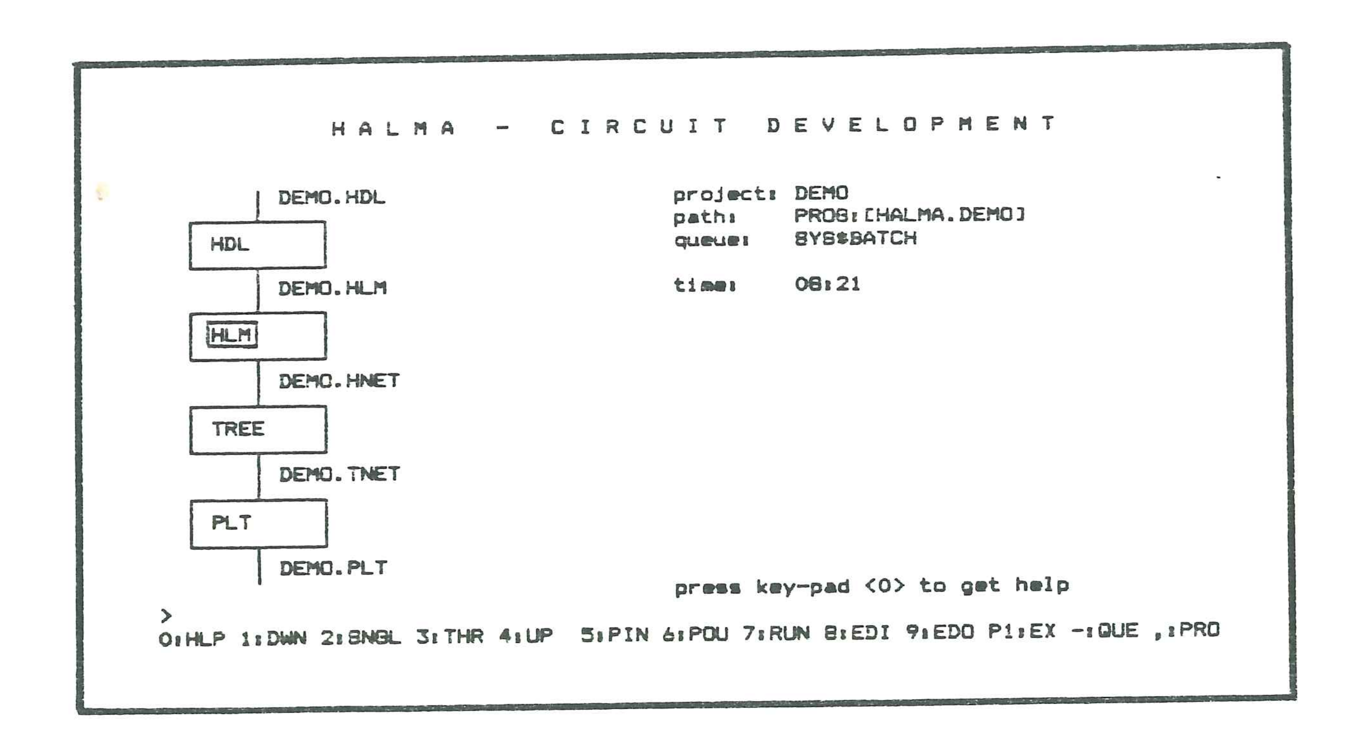
AUTOINSTALL aufrufen, wird HLMLOGIN.COM automatisch so angepaßt, daß die aktuelle Plattenkonfiguration berücksichtigt wird.

HLMLOGIN.COM überprüft, ob die für HLM$DISK vereinbarte physikalische Platte existiert und verwendet werden kann. Ist das nicht der Fall, so gibt HLMLOGIN.COM eine entsprechende Fehlermeldung aus und schlägt ein ASSIGN-Statement vor.

Bei der Installation muß auf jeden Fall per Hand oder mit AUTOINSTALL.COM die eine ASSIGN-Zeile in HLMLOGIN.COM überprüft und ggf. geändert werden.

HLMLOGIN.COM stellt fest, ob die PAGEFILE-QUOTA und das sog. BYTELIMIT groß genug sind. Im Fall von Fehlermeldungen sollten Sie Ihren VAX-Systemmanager bitten, Ihnen eine PAGEFILE-Quota von mindestens 30000 Blöcken und ein Bytelimit von mindestens 8000 einzurichten.

1.5 Das HALMA-Menü



HALMA tritt dem Benutzer über ein Bildschirm-Menü entgegen, das per DCL-Prozedur als Maske realisiert ist. Da zum Aufbau der Maske VT100-spezifische Escape-Sequenzen verwendet werden, läßt sich HALMA nur auf VT100-kompatiblen Terminals (z.B. VT101, VT102, VT220) starten.

Zur Menü-Auswahl werden die Keypad-Tasten rechts an der Tastatur verwendet. HALMA.COM setzt das Terminal in den sog. APPLICATION-Mode und kann so die Tasten direkt abfragen.

1.5.1 Hilfe!

H A L M A - H E L P

PF1 exit (twice)	PF2	PF3	PF4 submit job
7 run	8 edit input	9 edit output	- new queue
4 up	5 print input	6 print output	, new project
1 down	2 single step	3 through	

press <RETURN> to continue:

Die Belegung der Keypad-Tasten wird ständig in einer Fußzeile eingeblendet und kann außerrdem durch die "0"-Taste interaktiv erfragt werden (HELP-Funktion).

1.5.2 Programmteile auswählen

In der Maske werden schematisiert die vier Teilprogramme von HALMA dargestellt: HDL, HLM, TREE und PLT. Durch die Tasten "1" und "4" kann ein Masken-Cursor bewegt werden, der den gerade angewählten HALMA-Teil anzeigt. "4" (UP) bewegt den Cursor nach oben, "1" (DOWN) nach unten. Der Cursor wandert zyklisch weiter, sobald er einen Rand-Block durchschritten hat.

1.5.3 Starten eines Programmteils

Der mit dem Masken-Cursor ausgewählte Programmteil wird durch die RUN-Taste "7" gestartet. Als Ein- und Ausgabedateien werden die Dateien mit dem neben dem Block invers dargestellten Namen verwendet. Standardmäßig befindet sich HALMA im "single step mode", d.h. es wird nur ein Programmteil ausgeführt und dann angehalten. Durch Betätigen der THROUGH-Taste "3" schaltet man den single step mode aus, "2" (SINGLE) schaltet ihn ein.

1.5.4 Editieren der Eingabe

Durch die EDIT-INPUT-Taste "8" wird der Editor gestartet und man kann die Eingabedatei für den ausgewählten Programmteil ansehen oder editieren. Je nach angewähltem Programmteil wird der Editor anders initialisiert, so daß mit dem Editor-Kommando "GOLD H" stets ein passender Rahmen in die Eingabedatei kopiert werden kann. Diese Möglichkeit wurde für die Blöcke HDL und HLM eingebaut.

1.5.5 Ansehen der Ausgabe

Da die Ausgabe eines Blockes die Eingabe für den nächsten Block ist, kann man bis auf die Plot-Ausgabe die Ausgabedatei wie unter 1.5.4 beschrieben editieren, wenn man vorher den passenden Block angewählt hat. Direkt geht dies mit der EDIT-OUTPUT-Taste "9", die den Editor mit der Ausgabedatei des angewählten Blockes startet.

1.5.6 Der Projektname

Sämtliche Ein- und Ausgabedateien enthalten einen sogenannten Projektnamen wie "fulladder", "decoder" oder "alu74181". Mit Hilfe der NEW-PROJECT-Taste "," kann man einen neuen Projektnamen eingeben. Wird nur <RETURN> eingegeben, so bleibt der alte Name erhalten.

Folgende Dateinamen werden für jedes Projekt vereinbart:

project.HDL	Schaltnetzspezifikation
project.PRT	Protokoll des HDL-Compilers
project.HLM	Funktionstabelle im LOGE-Format
project.HNET	NAND2-Inverter Netzliste
project.TNET	lokal optimierte Netzliste
project.HILO	Netzliste im HILO-Format
project_1.PRN	Druck-Graphik von project.HNET
project_2.PRN	Druck-Graphik von project.TNET
project_1.PLT	Plot-Graphik von project.HNET
project_2.PLT	Plot-Graphik von project.TNET
project.LIS	LOG-File des Batch-Jobs

1.5.7 Stapelverarbeitung

Statt den angewählten Block interaktiv mit Hilfe der RUN-Taste "7" zu starten, kann man über die SUBMIT-Taste PF4 auch einen Batch-Job daraus machen. Einziger Unterschied zum interaktiven Betrieb ist, daß Batch-Jobs nicht im single step mode arbeiten sondern den ausgewählten und alle folgenden Blöcke ausführen. Das LOG-File des Batch-Jobs erhält den Namen "projectname.LIS". Mit Hilfe der NEW-QUEUE-Taste "-" kann man eine andere Batch-Queue wählen (<RETURN> ergibt die vorher gewählte Queue). Die Standard-Queue wird in der Maske angezeigt.

1.5.8 Ergebnisse ausdrucken

Durch Betätigen der PRINT-Tasten "5" oder "6" kann man Ergebnis-Dateien ausdrucken. "5" schickt die Eingabedatei des aktuell ausgewählten Blocks zum Drucken, "6" seine Ausgabedatei. Eine Ausnahme bildet der Block PLT. Hier wird die Druck-Graphik zum Drucker geschickt. Die Plot-Graphik kann nur mit Plot-Treiber-Programmen ausgegeben werden.

1.5.9 DCL-Kommandos zwischendurch

Da HALMA.COM ein DCL-Programm ist, lassen sich sämtliche DCL-Kommandos aus HALMA.COM heraus absetzen. So ist es z.B. möglich mit DIR die aktuelle Directory anzusehen, oder mit SET DEFAULT eine neue Directory anzuwählen. Man sollte natürlich nur solche Kommandos verwenden, die HALMA nicht "in's Gehege" kommen. Vorsicht ist insbesondere angezeigt bei Kommandos, die Dateidefinitionen verändern, Namen umbenennen, Protections setzen etc. Änderungen in der Datei SETUP.TREE werden z.Z. erst beim nächsten Aufruf von HALMA berücksichtigt.

Die aktuelle Directory wird stets oben rechts in der Maske angezeigt.

Durch das Kommando "HILONET" kann man zuvor durch TREE erzeugte Netzlisten in das HILO-Format für den Simulator HILO3 umwandeln.
Durch "SIMONET" erhält man eine Netzliste für den Simulator SIMON

1.5.10 Ausgabe von Plots

PLT erzeugt neben Druck-Graphiken auch PLOT-Graphiken (erkennbar an der Extension .PLT). Um Plot-Graphiken auszugeben, muß man passende Peripherie und passende Plot-Treiber verwenden. HALMA kann Tektronix-Terminals und Hewlett-Packard-Plotter ansteuern. Die PLT-Dateien haben ein intern am Institut für Theoretische Nachrichtentechnik verwendetes Vektor-Format, das sich leicht für alle möglichen Graphik-Geräte auswerten läßt.

Beispiel-Aufrufe:

```
@HLM$COM:TEKPLOT.COM project_2.PLT "1.0 0.0 0.0 0.0 1.0 0.0"

@HLM$COM:HPPLOT.COM project_1
```

Im ersten Beispiel wurden Transformationskoeffizienten angegeben, mit denen man vorgeben kann, wie der Plot skaliert werden soll. Die Koeffizienten sind:

mxx mxy bx mxy myy by

Die resultierende Abbildung berechnet sich nach den Beziehungen:

X' = mxx*X +mxy*Y +bx
Y' = mxy*X +myy*Y +by

Durch Angabe geeigneter Koeffizienten kann man den Plot also strecken, stauchen, spiegeln und drehen.

Die Plot-Treiber enthalten Hilfs-Informationen, die man wie folgt abrufen kann:

```
@HLM$COM:TEKPLOT ?

@HLM$COM:HPPLOT ?
```

1.5.11 HALMA beenden

Um HALMA.COM zu verlassen muß man zweimal die GOLD-Taste "PF1" betätigen oder LOGOUT eingeben.

1.5.12 Wenn ein Fehler eintritt

Fehler beim Ablauf von HALMA können prinzipiell zwei Gründe haben: Entweder ist die Benutzer-Eingabe fehlerhaft, oder Sie haben einen Fehler in HALMA aufgespührt.

Sollte das Letztere der Fall sein, so bitten wir um Nachricht mit dem Formular in Anhang D und entschuldigen uns für etwaige Frustrationen, die bei Ihnen durch HALMA entstanden sein sollten.

Bei Batch-Jobs geht die Fehlerursache meist aus dem LIS-File hervor. Interaktiv sind Fehlermeldungen manchmal schnell "verschwunden". Da bleibt dann nur ein weiterer Versuch per Batch-Job.

2 Übersetzung von Schaltnetzspezifikationen

2.1 Einführung

Um ein Schaltnetz zu spezifizieren, sind die folgenden Angaben notwendig:

- Wieviele Eingänge sind vorhanden?
- Wieviele Ausgänge?
- Welche Eingangskombinationen treten auf?
- Welche arithmetisch-logischen Beziehungen (Funktionen) soll das Schaltnetz realisieren?
- Gibt es Randbedingungen für die Verzögerungszeit?

Es wäre relativ unbequem, diese Angaben in einem festen Format, etwa in einer Tabelle, angeben zu müssen. Daher enthält HALMA einen Compiler, der in der Lage ist, Schaltnetzspezifikationen, die in einer PASCAL ähnlichen Spezifikationssprache abgefaßt wurden, formatfrei zu interpretieren und in einer Funktionstabelle umzuwandeln.

2.2 Die Spezifikationssprache

Eine Schaltnetzspezifikation für HALMA setzt sich aus mehreren Sektionen zusammen, die im folgenden beschrieben werden. Der Compiler ermittelt aus der Spezifikation, welche Eingangskombinationen auftreten können. Aus der Sektion SPECIFICATION erstellt er virtuellen Operations-Code, der von einer Stack-Maschine für jede vorkommene Eingangskombination abgearbeitet wird, um so die Funktionstabelle zu generieren.

Wichtig:

> Alle Zeilen einer Spezifikation dürfen erst ab Spalte 6 beschrieben sein, da sonst das Einfügen von Fehlermeldungen in den Quelltext nicht funktioniert!

Eine Sektion kann aus einer oder mehreren Anweisungen bestehen, die durch ";" getrennt werden. Ein "." schließt die Sektion ab.

Kommentare müssen in geschweifte Klammern eingeschlossen werden und dürfen nicht geschachtelt sein.

Groß- und Kleinschreibung sowie überflüssige Leerzeichen werden vom Compiler ignoriert. Statements dürfen beliebig über Zeilengrenzen gehen, wenn dadurch keine Bezeichner oder Sonderzeichen aufgespalten werden.

Bezeichnernamen dürfen zwar 30 Zeichen lang sein, man sollte sich jedoch auf weniger beschränken, da im LOGE-Format nur Namen mit 6 Zeichen zugelassen sind (HLM läßt 24 Zeichen zu). Da der HDL-Compiler aus Variablen mit mehr als einem Bit Wortlänge indizierte Variablennamen generiert, kann aus einem Namen mit 4 Zeichen im LOGE-Format ein Name mit 6 Zeichen werden. Namen müssen stets mit einem Buchstaben beginnen.

2.2.1 PROGRAM

Die Spezifikation beginnt mit einer Anweisung der Form:

PROGRAM name

Mit "name" wird das Schaltnetz zu Dokumentationszwecken bezeichnet.

Dahinter kann eine Anweisung der Form

(DEBUG = 1)

stehen, mit der man das Übersetzungsprotokoll steuern kann (siehe 2.3).

2.2.2 DECLARE: Variable vereinbaren

Die erste Sektion beginnt mit dem Schlüsselwort DECLARE. Sie muß sämtliche Eingangs-, Ausgangs- und Zwischenvariable mit den jeweiligen Wortlängen enthalten.

Beispiel:

```
DECLARE     a,b,cin: 1 BIT;
            out:     2 BYTE.
```

Nur die Standard-Variable IN darf nicht deklariert werden (siehe 2.2.5.4).

2.2.3 CONSTRAINTS: Wertebereiche festlegen

In der Sektion CONSTRAINTS wird festgelegt, welche Werte die Eingangsvariablen annehmen können. Dabei wird eine Intervallschreibweise verwendet:

Beispiel:

```
CONSTRAINTS   alpha: [1,3..5,7,9..2**4-1];
              b,cin: [0..1].
```

Wenn die in DECLARE vereinbarte Wortbreite das angegebene Intervall nicht aufnehmen kann, wird eine Fehlermeldung generiert. Variable mit demselben Wertebereich können in einer Liste aufgereiht werden. Negative Zahlen werden im 2er-Komplement dargestellt und brauchen daher für ihr Vorzeichen ein Bit mehr.

2.2.4 DELAYS: Laufzeiten vorgeben

In der Sektion DELAYS kann der Benutzer vorgeben, welche maximalen Verzögerungszeiten (in [ns]) das Schaltnetz aufweisen darf. Außerdem kann ein STARTDELAY angegeben werden, das aussagt, um wieviel eine Variable gegenüber den anderen verzögert am Eingang anliegt.

Beispiel:

```
DELAYS
  STARTDELAY: a   =  5;
  MAXDELAY:   out = 20.
```

2.2.5 SPECIFICATION: die eigentliche Beschreibung

Die Sektion SPECIFICATION enthält die funktionale Beschreibung des zu entwerfenden Schaltnetzes. Diese Sektion wird vom Compiler in eine Zwischenform umgesetzt, die für jede Eingangskombination abgearbeitet werden muß.

2.2.5.1 Wertzuweisungen

Für Wertzuweisungen wird wie in PASCAL das Zeichen ":=" verwendet, um es nicht mit dem logischen Vergleich zu verwechseln. Ist die Wortlänge des Zieloperanden größer als nötig, so wird mit Nullen aufgefüllt. Ist sie zu klein, so wird eine Overflow-Warnung generiert.

Die Zuweisung eines undefinierten Wertes führt zu einer Fehlermeldung.

2.2.5.2 Arithmetische Operationen

Außer den Grundoperationen +, *, / und - sind die monadischen Operationen

SQR(X) - Quadrat
SQRT(X) - Quadratwurzel
ABS(X) - Betrag
SHL(X) - um ein Bit nach links schieben
SHR(X) - um ein Bit nach rechts schieben

eingebaut worden. Wenn die Reihenfolge der Formelauswertung nicht durch Verwendung von Klammern festliegt, gilt "Punktrechnung geht vor Strichrechnung".

Die gewünschte Rechengenauigkeit in Bit kann hinter jedem Operator in eckigen Klammern angegeben werden:

a = b +[4] c;

Die Operationen werden in 2er-Komplement-Arithmetik unter Berücksichtigung der Operanden-Wortlängen durchgeführt.

Undefinierte Operanden führen zu einer Fehlermeldung.

2.2.5.3 Logische Operationen

An logischen Operationen sind AND, OR, NOT, und EXOR vorhanden, die ihre Operanden bitweise miteinander verknüpfen. Daneben gibt es die gewohnten Vergleichsoperatoren >, <, >=, <=, == und <>. Ergebnisse logischer Operationen dürfen nicht für arithmetische Operationen verwendet werden.

Eine besondere Operation ist die durch das Zeichen "//" dargestellte Konkatenation. Sie hängt zwei Operanden als Bitketten hintereinander.

2.2.5.4 IN und OUT

Die Funktionstabelle hat einen Eingangs- und einen Ausgangsteil. Mit Hilfe der Standard-Bezeichner IN und OUT muß der Benutzer vorgeben, in welcher Reihenfolge die Eingangs- und Ausgangsvariablen im jeweiligen Teil der Funktionstabelle aufgereiht werden sollen. Oft wird dazu der Konkatenations-Operator verwendet (siehe 2.2.5.3).

Beispiel:

```
IN:=a//b//cin;
OUT:=a+b+cin.
```

Werden IN oder OUT undefinierte Werte zugewiesen, so wird eine Fehlermeldung generiert. Don't Cares können nur mit Hilfe der Sektion CONSTRAINTS definiert werden.

Existiert nur eine einzige Eingangsvariable, so darf man **keine** Anweisung der Form

```
IN:=A;
```

schreiben. Die Zuweisung für **IN** hat lediglich den Zweck, die Reihenfolge der Variablen in der Funktionstabelle vorzugeben. Bei einer einzigen Variable steht die Reihenfolge ohne eine explizite Vorgabe fest.

2.2.5.5 IF .. THEN .. ELSE .. ENDIF

Die Spezifikationssprache erlaubt beliebig geschachtelte bedingte Anweisungen wie sie auch in anderen Programmiersprachen gebräuchlich sind.

Beispiel:

```
IF a<0 THEN
  a:=-1;
  sign:=-1
ELSE
  IF a==0 THEN
    sign:=0
  ELSE
    sign:=1
  ENDIF
ENDIF;
```

Innerhalb eines IF-Blocks sind aufeinanderfolgende Statements wieder durch ";" getrennt. Das Ende eines Blockes wird durch ein fehlendes ";" angezeigt.

2.2.5.6 CASE .. ENDCASE

Mit dem CASE-Konstrukt kann man bequem Fallunterscheidungen programmieren, wie sie z.B. bei Quantisierern auftreten. Im Gegensatz zum IF darf das CASE nicht geschachtelt werden; pro Fall ist nur ein Statement erlaubt.

Beispiel:

```
CASE
  a>0: sign:=+1;
  a==0:sign:=0;
  a<0: sign:=-1
ENDCASE;
```

Durch den "Fall" TRUE kann man erreichen, daß immer mindestens eine Bedingung erfüllt ist. Sind mehrere erfüllt, so wird die lexikalisch erste verwendet.

2.3 Das Übersetzungsprotokoll

Beim Auftreten von Fehlern kann es interessant sein, Details über die verwendeten Variablen und Statements zur Verfügung zu haben. Mit Hilfe eines Zusatzes der Form

(DEBUG = n)

(n = 0..3) kann man ein Übersetzungsprotokoll unterschiedlichen Umfangs bekommen. Mit (DEBUG = 0) schaltet das Protokoll ab, (DEBUG = 3) schaltet sämtliche Protokoll-Optionen an.

2.4 Editieren der Eingabe

Editiert man aus HALMA.COM heraus eine HDL-Eingabe, so kann man sich über das Editor-Kommando "GOLD H" einen vorgefertigten Rahmen für HDL-Dateien kopieren. Das erspart Schreibarbeit und entlastet das Gedächtnis in Bezug auf die HDL-Syntax.

2.5 Wenn ein Fehler eintritt

Stellt der HDL-Compiler Fehler fest, so werden diese an passender Stelle als Kommentare in den Quelltext gemischt. Alte Fehlerkommentare werden beim nächsten Lauf automatisch wieder gelöscht. Fehlerkommentare beginnen mit *ERR* und können daher im Editor leicht gefunden werden.

Wenn ein Fehler eintritt

Es kann in seltenen Fällen vorkommen, daß HDL durch Syntax-Fehler vollständig außer Tritt gerät und in einer Endlos-Schleife endet. Ein Benutzer-Abbruch durch Ctrl-Y führt dann wieder in das HALMA-Hauptmenü zurück. Man sollte jedoch nicht zu früh abbrechen, da sonst die letzte Version der HDL-Eingabe-Datei verlorengeht, wenn der Compiler beim Abbruch gerade dabei ist, alte Fehlerkommentare zu löschen oder Tabulator-Zeichen zu entfernen.

3 Synthese von unvermaschten NAND2-Inverter-Netzen

3.1 Einführung

Das HALMA-Teilprogramm HLM liest die Funktionstabelle des zu entwerfenden Schaltnetzes, minimiert sie und generiert dann ein unvermaschtes Schaltnetz aus Invertern und NAND-Gattern mit jeweils zwei Eingängen.

Diese NAND2-Inverter-Bäume werden dann als Netzliste an den nachfolgenden Block TREE (siehe Kapitel 4) zur weiteren Optimierung übergeben.

3.2 Die Eingabe-Syntax im LOGE-Format

Als Eingabe-Syntax wurde die Eingabe-Sprache des CAD-Programms LOGE von der Universität Karksruhe (Vertrieb: ISDATA GmbH, Karlsruhe) übernommen und mit einigen Erweiterungen versehen.

In den folgenden Abschnitten werden die einzelnen Bestandteile einer Eingabe für HLM beschrieben.

Vorweg noch einige grundsätzliche Hinweise:

Die HLM-Eingabe besteht aus mehreren Sektionen, die alle mit Schlüsselwörtern (erkennbar am "*") eingeleitet werden. Die Schlüsselwörter können ausgeschrieben oder soweit mit ihren Anfangsbuchstaben abgekürzt werden, daß sie noch eindeutig unterscheidbar sind (drei Buchstaben reichen).

Groß- und Kleinschreibung, überflüssige Leerzeichen sowie Tabulator-Zeichen werden ignoriert. Kommentare können entweder in einer "*COMMENT"-Sektion untergebracht oder als Inline-Kommentar mit einem ";" eingeleitet werden.

Die Reihenfolge der Sektionen ist prinzipiell beliebig, soweit einzelne Sektionen nicht Informationen aus anderen Sektionen voraussetzen. Wer die Sektionen in der Reihenfolge verwendet, wie sie im folgenden beschrieben werden, macht nichts verkehrt.

3.2.1 Deklaration der Schaltnetzparameter

Die drei Sektionen "*DECLARATIONS", "*X-NAMES" und "*Y-NAMES" geben die Anzahl und die Namen der Eingangs- und Ausgangsvariablen an.

Ein Beispiel vorweg:

```
*DECLARATIONS
  X-VAR = 4      ; n
  Y-VAR = 7      ; m
*X-NAMES
  DAT1=1,DAT2=2,RST=3,CLK=4;
*Y-NAMES
  OUT1=7,OUT2=6,OUT3=5,OUT4=4;
  OUT5=3,OUT6=2,OUT7=1;
```

Wenn die Variablennamen nicht explizit angegeben werden, nimmt HLM die Standard-Namen X1, X2, ... Xn sowie Y1, Y2, ... Ym an. Die Variablen werden in der Funktionstabelle von links nach rechts beginnend mit 1 durchnumeriert. Variablennamen dürfen bis zu 24 Zeichen lang sein. Im Original-LOGE sind nur sechs Zeichen zugelassen.

Mit dem Schlüsselwort "*IDENTIFICATION" kann man den Entwurf mit einem Namen versehen:

```
*IDENTIFICATION
  decoder
```

Durch Verwendung des Schlüsselwortes "*INFORM" kann man Informationen über die HALMA-Eingabe erhalten. Die HALMA-Optionen werden erläutert. Außerdem werden die Heuristik-Parameter und die gewählten Optimierungs-Modi ausgegeben.

3.2.2 Die Funktionstabelle

Die Funktionstabelle beginnt mit dem Schlüsselwort "*FUNCTION-TABLE" und endet vor dem nächsten Schlüsselwort.

Ein Beispiel vorweg:

```
*FUNCTION-TABLE
  0000 101010X
  0001 XXXXXX1   ; entspricht 0001 ------1
  X011 0000111
  REST XXXX000   ; für alle anderen Eingangskombinationen
* ...
```

Der Eingangsteil jeder Zeile muß n '0'-, '1'- oder 'X'-Trits (Trit = ternary digit) enthalten und darf beliebig durch Leerzeichen, Doppelpunkte oder Kommata unterbrochen werden. Entsprechend muß der Ausgangsteil m Trits enthalten. Für Don't Cares kann statt 'X' auch '-' verwendet werden.

Die abschließende REST-Zeile ist obligatorisch. Sie gibt die Ausgangskombination für nicht spezifizierte Eingangskombinationen an und sollte möglichst "REST XXX..X" lauten, um dem Optimierungsprozeß alle Freiheiten zu lassen.

3.2.3 Formeleingabe

Man kann statt einer Funktionstabelle die zu realisierenden Schaltfunktionen auch formelmäßig angeben.

Für jede Ausgangsvariable muß dann ein Boolescher Ausdruck in der Sektion "*TRANSLATE" angegeben werden. Zur Zeit dürfen die folgenden Operatoren in den Ausdrücken verwendet werden:

!	Negation
^	Exclusive-OR
!^	Exclusive-NOR
*	AND (auch durch Konkatenation)
+	OR

!+ NOR
| NAND

Beispiel:

```
*TRANSLATE
  Y1=X1*X2;        entspricht Y1=X1X2; oder Y1=X1 X2;
  Y2=X1^(X2+!X3);
```

Mit Hilfe von runden Klammern kann die Abarbeitungsreihenfolge beliebig vorgegeben werden. Ansonsten gelten die Prioritätsregelungen, wie sie in der Booleschen Algebra üblich sind.

Als Variablen-Namen müssen X1 ..Xn und Y1 .. Ym oder, falls per *X-NAMES und *Y-NAMES definiert, die entsprechenden eigenen Namen verwendet werden.

3.2.4 Optimierungskriterien

Mit der Sektion "*OPTIMIZE" können verschiedene Optimierungsarten ausgewählt werden.

Zur Zeit sind folgende Arten eingebaut:

DELAY - auf minimale Stufenzahl optimieren
GATES - Gatterzahl minimieren (DELAY geht vor GATES)
URGENT - möglichst schnell eine Lösung suchen (auf Kosten der Lösungsgüte)
RAPID - eine Art Super-URGENT mit entsprechenden Konsequenzen.

Zunächst sollte man mit DELAY anfangen. Wenn die Rechenzeit kurz war und die erzielte Gatterzahl zu hoch, ist ein zweiter Versuch mit der Kombination DELAY,GATES empfehlenswert. Hat man es ganz eilig oder möchte nur einen Überblick gewinnen, so nehme man URGENT oder gar RAPID.

Beispiel:

```
*OPTIMIZE
  GATES,DELAY,RAPID;
```

3.2.5 Heuristik-Parameter

Mit vier Heuristik-Parametern kann der Optimierungsprozeß entscheident beeinflußt werden:

RDEPTH gibt an, bis zu welcher Stufenzahl Lösungen gesucht werden sollen. Je kleiner RDEPTH gewählt wird desto weniger Rechenzeit benötigt HLM. Gleichzeitig steigt aber auch das Risiko, gar keine Lösung zu finden. In der Ausgabe von HLM findet man Angaben darüber, welche Suchtiefe wahrscheinlich ausreicht.

CUBLIM, CUBLIMH und CUBLIMT steuern die angewendeten Zerlegungsstrategien. Sie geben an bis zu welcher Größe von Unterfunktionen welche Zerlegungsmethode verwendet wird.

Praktische Werte sind:

```
*HEURISTICS
  CUBLIMH = 5     ; zwischen 1 und 8 wählen
  CUBLIMT = 3     ; für CUBLIM=99 ziwschen 1 und 6 wählen
  CUBLIM  = 7     ; größer als CUBLIMH aber kleiner als 10
  RDEPTH = 10     ; erlaubte Stufenzahl
```

Der Wert von CUBLIMH muß immer unter dem von CUBLIM liegen, je größer beide sind, umso besser wird die Lösung und umso mehr Rechenzeit wird benötigt.

CUBLIMT kommt erst dann zum Tragen, wenn CUBLIM auf 99 gesetzt wurde. In diesem Fall ist CUBLIMH irrelevant.

3.2.6 Synthese-Arten

HLM enthält verschiedene Optimierungsalgorithmen, die über entsprechende Schlüsselwörter ausgewählt werden können.

Übersicht:

```
*MULTI-LEVEL     ; mehrstufiger Entwurf, Default: zweistufig
*SINGLE          ; Einzeloptimierung,    Default: Bündeloptim.
*SIMPLIFY        ; schnell aber suboptimal
*PRESS           ; Zeilen mit gleichem Ausgangsteil vorher
                 ;   zusammenfassen
```

Mit einer "*POLARITY"-Sektion kann man ohne Änderungen in der Funktionstabelle oder in der Formeleingabe dafür sorgen, daß bestimmte Ausgänge invertiert, normal oder gar nicht berechnet werden. Der Schaltungsaufwand läßt sich durch Invertieren manchmal senken.

Beispiel:

```
*POLARITY
  ORIGINAL  DATA1,DATA2,FIFO;
  INVERTED  RST;
```

Nicht angegebene Signale werden auch nicht berechnet. Ohne *POLARITY-Sektion werden alle Ausgänge normal berechnet (=ORIGINAL).

3.2.6.1 Zweistufige oder mehrstufige Synthese?

Im Fall einer zweistufigen Optimierung wird lediglich die Funktionstabelle zusammengefaßt und kein mehrstufiges Schaltnetz erzeugt. Es kann sinnvoll sein, zunächst nur zweistufig zu optimieren. Erstens kann man dann entscheiden, ob ein PAL/PLA-Baustein ausreicht, und zweitens kann man das Ergebnis für nachfolgende mehrstufige Optimierungen verwenden und so Rechenzeit einsparen.

Die Mehrstufen-Optimierung wird durch "*MULTI-LEVEL" eingeschaltet.

3.2.6.2 Bündel- oder Einzel-Optimierung?

Bei einer Bündeloptimierung wird versucht, alle Schaltfunktionen gleichzeitig und unter gegenseitiger Berücksichtigung zu optimieren. Das braucht im Normalfall mehr Rechenzeit und führt zu Lösungen, die weniger Gatter aber oft größere Delays aufweisen.

Mit "*SINGLE" kann man die Bündeloptimierung abschalten. Dann wird für jeden Ausgang einzeln optimiert und erst in TREE wieder zusammengefaßt. Diese Betriebsart ist schneller und führt zu Lösungen, die kürzere Delays aber oft mehr Gatter benötigen.

3.2.7 Im Falle von Eingabefehlern

Syntax-Fehler werden von HLM mit Angabe der fehlerhaften Zeile (oder der unmittelbar folgenden Zeile) ausgegeben. Die meisten Fehler füren zu einem Programm-Abbruch. Lediglich nach informativen Warnungen wird weitergerechnet.

3.3 Bestandteile der Ausgabe

Die Ausgabe von HLM enthält neben einer Kopie der Eingabe Schätzungen für die Stufenzahl der Lösung, eine Laufzeitstatistik, etwaige Fehlermeldungen und eine Zusammenfassung für den schaltungstechnischen Aufwand der gefundenen Lösung.

3.4 Die Syntax der erzeugten Netzliste(n)

Die erzeugte Netzliste besteht aus einer Zeile pro Gatter und enthält außerdem Angaben über die Variablennamen.

Eine Beipielnetzliste vorweg:

```
N1=NAND2(X1,X2)
N2=INV(X1)
N3=INV(X2)
N4=NAND2(N2,N3)
N5=NAND2(N1,N4) "A1"
```

```
X1=A
X2=B
A1=A^B
END
```

Die Knoten werden mit N1 .. Nk bezeichnet. Eingangsvariable heißen X1 .. Xn. Die folgenden Gattertypen sind vorhanden:

AND
NAND
OR
NOR
EXOR
EXNOR
INV

Bei Gattern mit mehr als einem Eingang wird die Zahl der Eingänge dem Namen angehängt, wie im Beispiel zu sehen ist.

Wird ein Eingang direkt zu einem Ausgang des Schaltnetzes durchgeschaltet, so erzeugt HALMA z.Z. statt eines "Drahtes" zwei hintereinandergeschaltete Inverter. Diese Notlösung wurde gewählt, um ohne den "Gatter"typ WIRE auszukommen.

Nach dem END-Statement dürfen beliebig viele weitere Netzlisten folgen. Die genaue Syntax finden Sie in Anhang C.

3.5 Wenn HLM keine Lösung findet

Wenn HLM keine Lösung gefunden hat, liegt meist einer der folgenden Gründe vor:

1. Die Suchtiefe RDEPTH wurde zu niedrig vorgegeben
2. Die Heuristik-Parameter CUBLIM, CUBLIMH und CUBIMT wurden nicht groß genug angegeben.
3. Wegen der Optimierungs-Modi URGENT oder RAPID wurde keine Lösung gefunden.

Abhilfe schafft dann ein neuer Versuch mit "großzügiger" angegebenen Parametern.

3.6 Im Fehlerfall

Kommt HLM zu keiner Lösung oder bricht gar mit einer Fehlermeldung ab, so bleibt nur folgendes zu tun:

1. Die Installation von HALMA überprüfen: Fehlende oder falsch definierte Dateien und Symbole können zum Abbruch führen.
2. Die User-Quotas überprüfen: Wenn die Disk-Quota oder die Page-File-Quota zu niedrig sind, kann es zu Fehlern kommen.
3. Mit geänderten Parametern weiter"probieren".
4. Das Formular in Anhang D ausfüllen und einsenden.

4 TREE: Optimierung durch lokale Transformationen

4.1 Einführung

Die von HLM erzeugten unvermaschte NAND2-Inverter-Schaltnetze lassen sich meist noch auf zwei Arten verbessern:

1. Durch sogenannte "lokale Transformationen", bei denen Teilschaltnetzte durch Gatterkonstellationen ersetzt werden, die günstigere Eigenschaften aufweisen.
2. Durch Zusammenfassen äquivalenter Unterschaltnetze.

Diese beiden Optimierungsmethoden wendet das HALMA-Teilprogramm TREE an, nachdem es die zuvor erzeugte Netzliste gelesen und als Graph im Rechner abgelegt hat.

Die Auswahl der auszuführenden lokalen Transformationen wird durch eine Bewertungsfunktion gesteuert, die angibt, wie "gut" das Gesamtschaltnetz vor und nach der Transformation war bzw. ist.

TREE überprüft, ob durch die Transformationen das logische Verhalten des Schaltnetzes verändert wurde. Ist dies der Fall, so liegt normalerweise ein Programmfehler vor, der dann durch eine entsprechende Meldung angezeigt wird.

4.2 Drei Optimierungs-Modi

TREE enthält drei Strategien, um die nächste lokale Transformation unter allen anwendbaren Transformationen auszuwählen:

1. Jede anwendbare Transformation, die die Bewertungszahl des Schaltnetzes erhöht, wird ausgeführt.
2. Jede anwendbare Transformation, die die Bewertungszahl des Schaltnetzes um eine vorgegebene Schwelle erhöht, wird ausgeführt. (Dieser Modus ist z.Z. nicht eingebaut)
3. Von allen anwendbaren Transformationen wird diejenige ausgeführt, die die Bewertungszahl am meisten erhöht.

Die dritte Methode führt im allgmeinen zu den besten Ergebnissen, braucht aber auch die meiste Rechenzeit. Die zweite Methode enthält die erste als Sonderfall (Schwelle 0) und eignet sich für Fälle, in denen schnell eine Lösung ermittelt werden soll.

Durch Editieren der Datei SETUP.TREE kann der Benutzer den Optimierungsmodus von TREE verändern.

4.3 Gewichtung der Optimierungskriterien

Die Bewertungsfunktion besteht aus einer gewichteten Summe der vier Schaltungsparameter Gatterzahl, Verbindungszahl, Transistorzahl und Verzögerungszeit:

B = -(Gg*Gates +Gc*Connects +Gt*Transistors +Gd*Delay)

Durch Wahl der Gewichtskoeffizienten kann der Benutzer entscheiden, ob mehr auf Gatterzahl (entspr. Fläche) oder auf Verzögerungszeit optimiert werden soll.

Das negative Vorzeichen sorgt dafür, das die Bewertungszahl maximiert werden muß und nicht minimiert.

Die Koeffizienten entnimmt TREE der Datei GDVM.DAT auf der aktuellen Benutzer-Directory. Dort können sie per Editor verändert werden.

4.4 Eingabe der Technologie-Parameter

Außer den Gewichten enthält GDVM.DAT noch die Technologie-Parameter der verwendbaren Gattertypen:

Für jeden Gattertyp:

- Anzahl der Transistoren
- Delay in Abhängigkeit vom Fan-Out
- Fan-In

Das Delay wird durch je zwei Koeffizienten angegeben:

$$D = k_1 + k_2 * FO$$

D - Delay in [ns]

k_1 - Verzögerngszeit bei FO=0, d.h. die Zeit um eigene Kapazitäten umzuladen

k_2 - Faktor, mit dem FO beim Delay eingeht, d.h. die Zeit um eine Standardkapazität am Ausgang umzuladen

FO - Fan-Out in Inverter-Äquivalenten

Soll TREE mit einer neuen Zellbibliothek verwendet werden, so muß GDVM.DAT entsprechend geändert werden.

4.5 Steuerung der Protokoll-Ausgabe

Neben dem Optimierungs-Modus kann der Benutzer vorgeben, ob jede untersuchte Transformation auf dem Bildschirm ausgegeben werden soll. Dies kostet zwar Rechenzeit, gibt aber Auskunft über den Fortgang der Optimierung. Um das Protokoll ein- oder auszuschalten, muß man die Datei SETUP.TREE editieren.

4.6 Im Fehlerfall

Fehler beim Gebrauch von TREE können sein:

1. Syntax-Fehler in der Netzliste oder fehlende Netzliste
2. Syntax-Fehler in GDVM.DAT oder GDVM.DAT nicht vorhanden
3. Programm-Fehler

Während man Syntax-Fehler dank der entsprechenden Meldungen meist recht schnell beheben kann, hilft im dritten Fall nur der Griff zum Formular für Fehler-Reports in Anhang D.

Manchmal hilft auch ein neuer Versuch mit einem anderen Optimierungs-Modus, doch noch zu einer Lösung zu kommen.

5 PLT: Plot-Ausgabe von Schaltnetzen

5.1 Einführung

Um die von HALMA entworfenen Schaltnetze nicht nur als Netzliste sondern auch als Schaltplan zur Verfügung zu haben, enthält HALMA eine graphische Ausgabe. Das Teilprogramm PLT liest Netzlisten ein und gibt sie als Druck-Graphik oder als Plot-Graphik aus.

5.2 Zwei Normen für Schaltzeichen

PLT kann Schaltpläne sowohl in der vielen Entwicklern vertrauten "Halbkäse"-Darstellung als auch in der neuen DIN-Darstellung ausgeben. Die Druck-Graphik verwendet stets die kastenförmigen Gattersymbole nach DIN.

Folgende Varianten von PLT stehen zur Verfügung:

1. HALMAPLOT stellt in der Plot-Graphik Gatter als "Halbkäse" dar und erzeugt parallel eine Druck-Graphik
2. HALMADRUCK erzeugt keine Plot-Graphik sondern nur eine Druck-Graphik
3. HALMAPLOTNEU erzeugt Plot- und Druck-Graphik in Kästchen-Darstellung nach DIN

5.3 Druck- oder Plot-Ausgabe

Die Druck-Graphik läßt sich als normale ASCII-Datei auf Drucker oder Terminal ausgeben. Ist das Schaltbild so breit, daß es nicht in 132 Zeichen paßt, so werden mehrere Streifen (bis zu drei) erzeugt, die dann nebeneinandergelegt werden müssen. Druck-Graphik-Dateien haben die Extension ".PRN".

Plot-Graphik-Dateien werden in einem Vektor-Format beschrieben, das intern am Institut für Theoretische Nachrichtentechnik verwendet wird. Wer die zugehörige Plot-Software nicht zur Verfügung hat, kann die auf dem HALMA-Band befindlichen Plot-Treiber zur

Ausgabe der Graphik benutzen.

Plot-Graphik-Dateien können recht umfangreich werden; sie haben die Extension ".PLT".

In den von PLT erstellten Graphiken gibt es manchmal außer normalen Leitungen auch solche, die durch Punkt-Linien dargestllt sind. Hier handelt es sich um sogenannte Komplexgatter. Das sind Gatterkonstellationen, die sich durch spezielle Transistor-Schaltungen besser realisieren lassen, als wenn man sie aus konventionellen Gattern aufbaute. Im Schaltplan wird in solchen Fällen die äquivalente Konstellation konventioneller Gatter dargestellt, die zur Unterscheidung mit punktierten Leitungen verbunden sind.

5.4 Ausgabe auf Tektronix-Terminals

Mit Hilfe der Programme TEKPLOT.COM und TEKPLOT.EXE können HALMA-Plots auf Tektronix-Terminals ausgegeben werden. Die Prorgamme wurden in Hannover mit dem Tek 4115b und dem VT100RG-Terminal erprobt.

Bedienungshinweise erhält man mit dem Aufruf:

```
@HLM$COM:TEKPLOT ?
```

5.5 Ausgabe in HP-GL

Ähnlich wie TEKPLOT HALMA-Plots auf Tektronix-Terminals ausgibt, wandelt HPPLOT Dateien vom HALMA-Plotformat in die Hewlett-Packard Graphics Language (HP-GL) um. Die umgewandelten Dateien können dann auf HP-GL-fähigen Plottern ausgegeben werden.

Nähere Bedienungshinweise ergibt der Aufruf:

```
@HLM$COM:HPPLOT ?
```

5.6 Im Fehlerfall

Drei Arten von Fehlern können bei der Benutzung von PLT auftreten:

1. Syntax-Fehler in der Netzliste
2. Überlauf der PLT-internen Arbeitsblatt-Matrix
3. Programm-Fehler

Nur die erste Art von Fehler kann von Benutzer selbst behoben werden. Bei einem Überlauf war die auszugebende Schaltung so groß, daß PLT recompiliert werden müßte um sie ausgeben zu können. Programmfehler sollten Sie uns mit dem Formular in Anhand D mitteilen.

6 Literatur zu HALMA

Biehl,G.;Ditzinger,A.: **"LOGE Benutzerhandbuch".** ISDATA GmbH, Karlsruhe,(1984).

Dietmeyer,D.L.: **"Logic Design of Digital Systems".** Allyn and Bacon, Boston,(1978).

Frank,Th.: **"Rechnerprogramm zur graphischen Darstellung von Schaltnetzen".** Studienarbeit am Institut für Theoretische Nachrichtentechnik der Universität Hannover,(1986).

Kemper,A.;Sarfert,Th.M.: **"HALMA: ein CAD-Programm für den vollautomatischen Entwurf mehrstufiger Schaltnetze".** Beitrag auf dem 2. E.I.S.-Workshop, GMD Bonn (1986), nachgedruckt in: Informationstechnik - IT,Vol.28,Nr.3, Oldenbourg Verlag, München, Juli (1986), pp.138-141.

Kemper,A.: **"Vollautomatische Schaltnetzsynthese unter Berücksichtigung technologischer und anwendungssprezifischer Randbedingungen."** Als Dissertation eingereicht, Universität Hannover, Institut für Theoretische Nachrichtentechnik, Juni(1986).

Krönke,B.: **"Entwurfssprache für Schaltnetze und Implementation des zugehörigen Übersetzers".** Studienarbeit am Institut für Theoretische Nachrichtentechnik der Universität Hannover,(1986).

Sarfert,Th.M.: **"Optimierung von Schaltnetzen durch lokale Transformationen unter Berücksichtigung technologischer Randbedingungen".** Diplomarbeit am Institut für Theoretische Nachrichtentechnik der Universität Hannover, (1985).

Im Herbst 1986 soll zu HALMA noch ein BMFT-Abschlußbericht veröffentlicht werden.

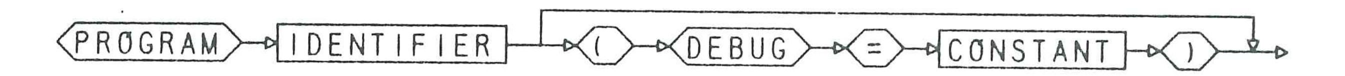
Anhang A:

Syntaxgraphen

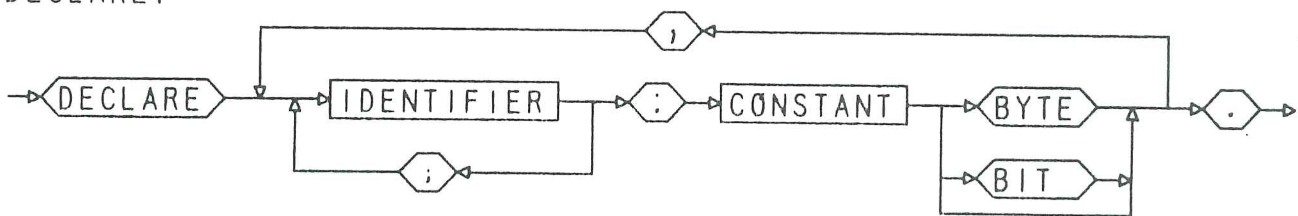
PROGRAMMAUFBAU:



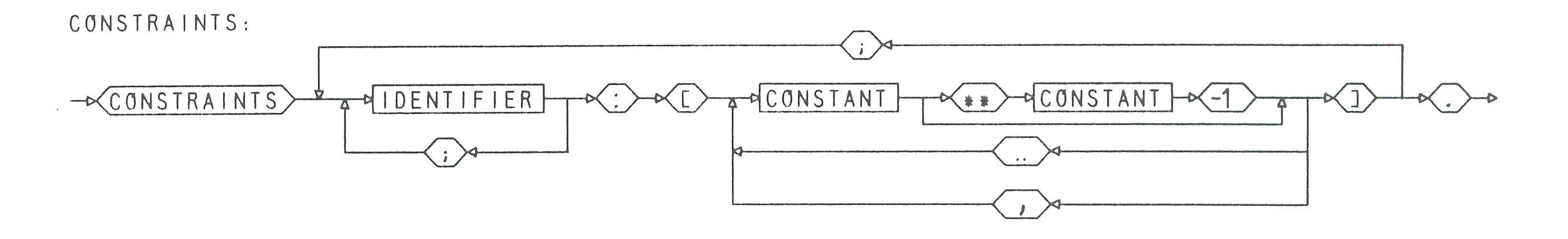
HEADER:



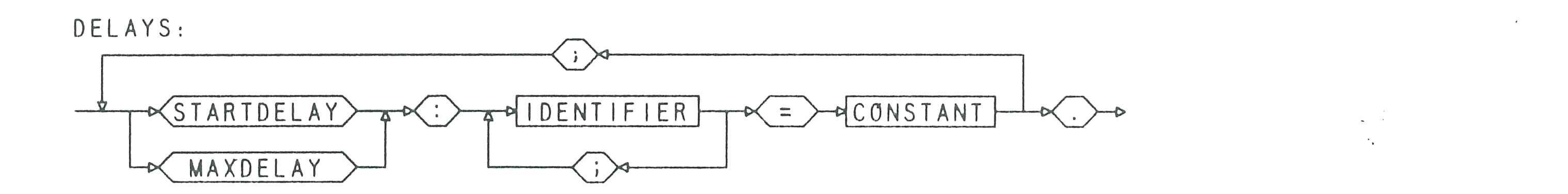
DECLARE:

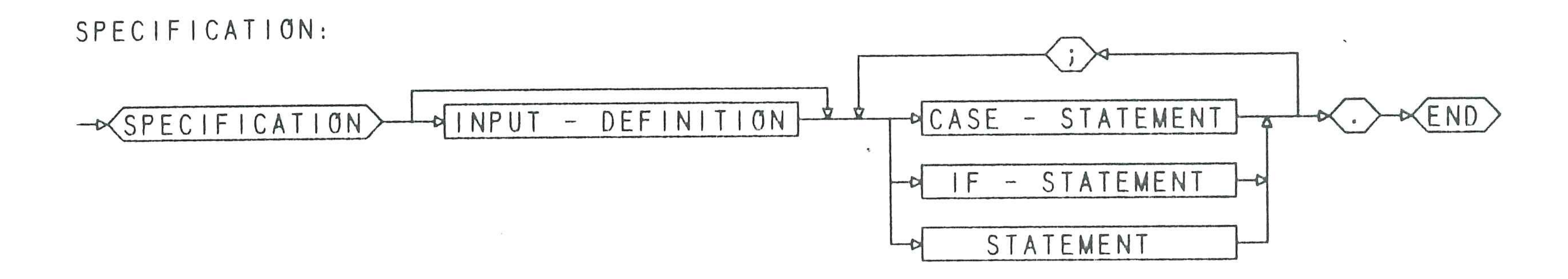


CONSTRAINTS:

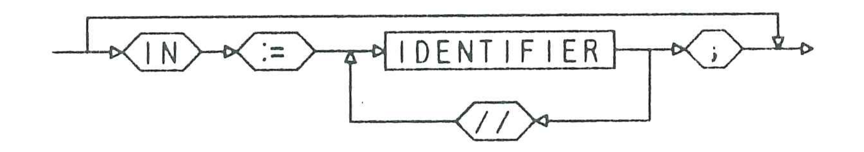


DELAYS:





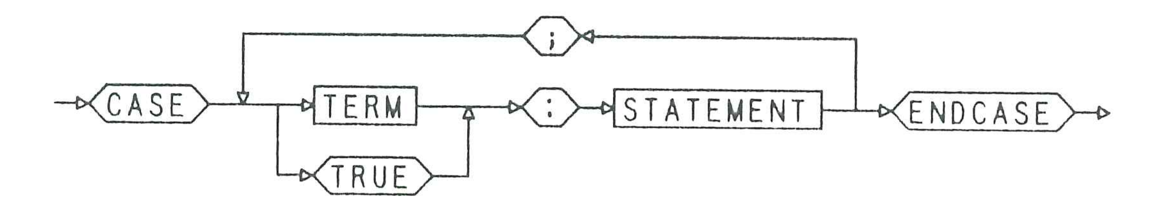




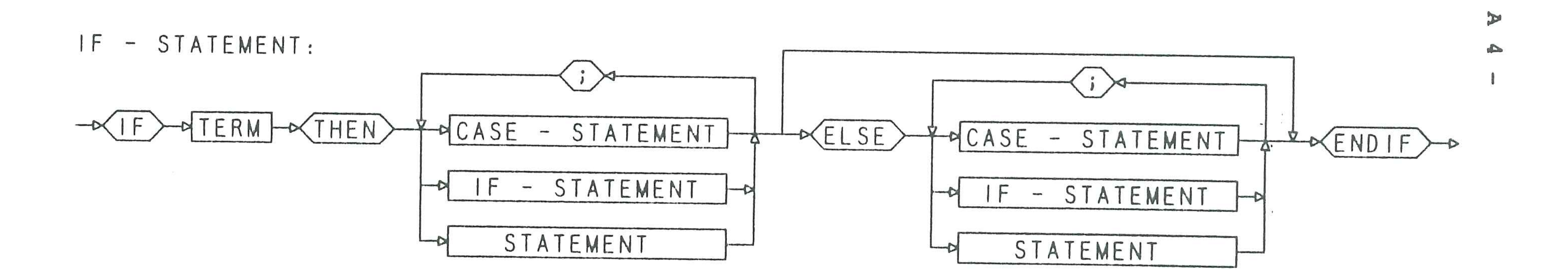
STATEMENT:

IDENTIFIER → := → TERM

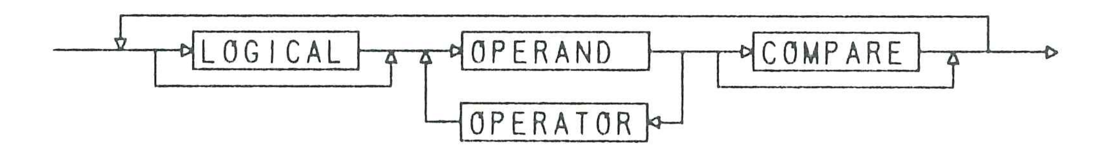
CASE - STATEMENT:



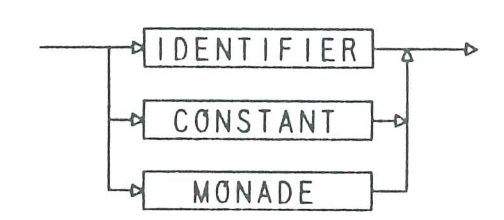
IF - STATEMENT:



TERM:



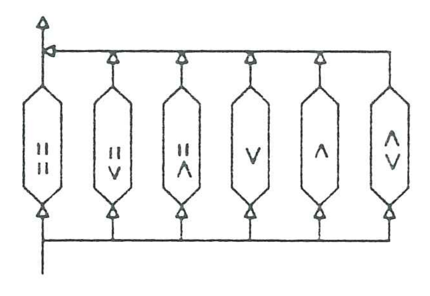
OPERAND:



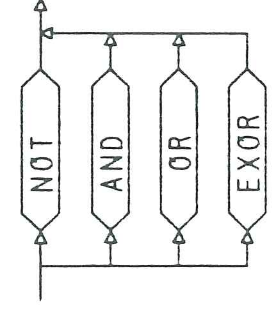
MONADE:

MONADE - OPERATOR → (→ OPERAND →) →

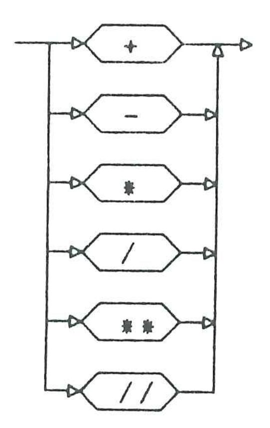
COMPARE:



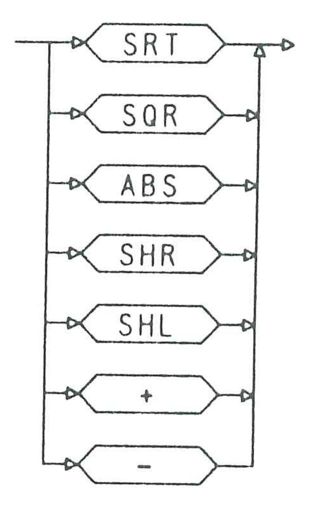
LOGICAL:



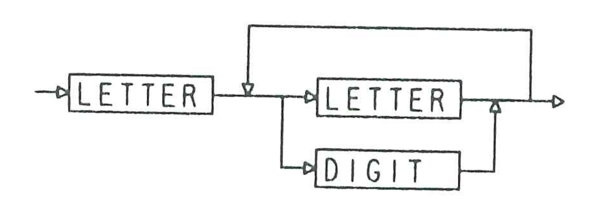
OPERATOR:



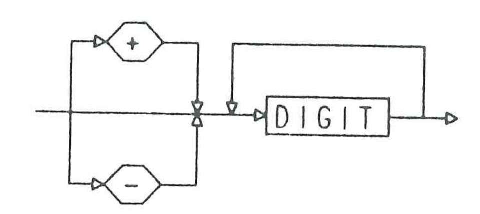
MONADE - OPERATOR:



IDENTIFIER:



CONSTANT:



Die Syntax von HLM

Die HLM-Syntax ist ein SUPERSET der LOGE-Syntax, d.h. HLM sollte jede korrekte Eingabedatei für LOGE-SNE, LOGE-PLA und LOGE-RAM verarbeiten können. Schlüselwörter fangen mit '*' an, Kommentare werden mit ';' eingeleitet.

Der Text bis zum ersten Schlüsselwort wird als Kommentar überlesen.

Leerzeilen und Blanks können beliebig eingestreut werden. Die Reihenfolge der Eingabesektionen ist nur dadurch eingeschränkt, dass vor der Funktionstabelle feststehen muss, wieviele Variable vorliegen.

Tauchen Sektionen doppelt auf, so gilt die jeweils letzte. Nicht aufgeführte LOGE-Schlüsselwörter werden ignoriert (z.B. *PARTITION, *RAM, *PLA etc.).

Wenn nichts anderes angegeben wurde, verhält sich HALMA wie LOGE-PLA: es wird eine zweistufige Bündel-Minimierung mit MOMIN durchgeführt. Alle in dem Syntax-Demonstrations-beispiel genannten Flags sind ausgeschaltet.

Die Ausgabe von HALMA stimmt nicht mit der von LOGE überein. Ausgegeben werden entweder die Netzliste der mehrstufigen Lösung oder eine Funktions-Tabelle der minimierten zweistufigen Lösung.

HLM - Syntax:

```
*IDEntification          ; bei Schlüsselwörtern reichen 3 Buchstaben
<name bis 40 Zeichen>
*HEUristics
CUBLIMH=5                ; Cube-Limit für HALFS
CUBLIMT = 8              ; Cube-Limit für TREESPLIT
CUBLIM = 7               ; Cube-Limit für GENKLST
RDEPTH=5                 ; maximale Rekursionstiefe für BEST1ST
*OPTIMIZE                ;
URGENT,GATES,DELAY,RAPID;
```

```
*SINgle                       ; jeden Ausgang für sich optimieren
*MULti-level                  ; mehr als zwei Stufen möglich
*PREss                        ; schaltet PRESS vor die Optimierung
*SIMplify                     ; verwendet SIMPLIFY
*INFORM                       ; Ausgabe von Informationen über d. Eingabe
*DEClarations
X-Var = 6                     ; maximal 64
Y-Var = 4                     ; maximal 32
CPU-time = 1000               ; wird von HALMA ignoriert
*X-Names
I1=1,I2=2,I3=3,I4=4,
I5=5,I6=6;                    ; Default-Namen X1, X2 ...
*Y-Names
OUT1=1,OUT2=2,OUT3=3,
OUT4=4;                       ; Variablen-Namen bis zu 24 Zeichen lang
*COMment
<beliebiger Kommentar bis zum nächsten Schlüsselwort>
*URGent                       ; schaltet das URGENT-Flag ein
*CONsist                      ; Tabelle wird auf Konsistenz geprüft
*REDundant                    ; Tabelle wird auf Redundanz geprüft
*FUNction-table
110010 00X1                   ; in die Zeilen dürfen Blanks und Kommas
001100 1110                   ; eingestreut werden
X01XX1 X1X1
X10XX1 0000
REST   ----                   ; REST muß immer (!!) angegeben werden
```

statt (!) *FUNCTION:

```
*TRANSLATE
  OUT1 = I1^I2;               ; BOOLEsche Ausdrücke
  OUT2 = (I1 !I3) |
         (I3 !+ I2);

*POLARITY                     ; Ausgänge können invertiert u.
ORIGINAL OUT1,OUT2;           ;     ignoriert werden
INVERTED OUT3;                ; OUT4 wird nicht berechnet!
*END
```

Syntax der HALMA-Netzlisten

```
              netlists::= netlist "END" [netlists]
              netlist ::= statement <statement> <inputdeclaration>
                          <outputdeclaration>
            statement ::= nodenum " = " ["C"] gatetype [incount]
                          "(" nodenum | var <"," nodenum | var> ")"
                          [comment [comment]]
              nodenum ::= ("N"|"G") number
             gatetype ::= "INV" | "NAND" | "NOR" | "AND" | "OR" |
                          "EXOR" | "XNOR"
              incount ::= number | ""
                  var ::= "X" number
              comment ::= """ [char [char [char]]] """ |
                          """ "A" number """ |
                          """ "C" number """
     inputdeclaration ::= var "=" identifier
    outputdeclaration ::= "A" number "=" identifier
           identifier ::= char <char | digit>
               number ::= digit <digit>
                digit ::= "0" | "1" ... | "9"
                 char ::= "a" ..| "z" | "A" ..| "Z"
```

Groß- und Kleinschreibung wird nicht unterschieden. Leerzeichen dürfen außer in Schlüsselwörtern beliebig eingeschoben werden. Statements dürfen nur bis zu eine Zeile lang sein. Die Anzahl der Eingangs-Parameter muß mit dem jeweiligen "INCOUNT" übereinstimmen. "INCOUNT" darf bei Invertern weggelassen werden. "C" vor einem Gattertyp (z.B. CNAND2()) bedeutet, daß das Gatter Teil eines Komplexgatters ist.

Fehler-Meldeformular:

Bitte einsenden an:

Universität Hannover
Institut für Theoretische Nachrichtentechnik
und Informationsverarbeitung
A.Kemper
Callinstr. 32/1305
3000 Hannover 1

Name:

Institution:

Adresse: Telephon:

Rechenanlage: HALMA-Version:

Fehler-Beschreibung:

Bitte legen Sie Ihrem Fehler-Report möglichst aussagekräftige Informationen (die verwendeten Benutzer-Dateien, Trace-Back-Dumps etc.) bei.

Vielen Dank für Ihre Mitarbeit!

Listing of save set(s)

6250 BPI Aufzeichnungsdichte

```
Save set:            HALMA.BKP
Written by:          KEMPER
UIC:                 [070,002]
Date:                12-SEP-1986 09:27:49.21
Command:             BACKUP/VERIFY/REW [.HALMA...] MTA0:HALMA.BKP
Operating system:    VAX/VMS version V4.3
BACKUP version:      V4.3
CPU ID register:     013811B5
Node name:           _TNTHAN::
Written on:          _MTA0:
Block size:          8192
Group size:          10
Buffer count:        3

[KEMDIR.HALMA]COM.DIR;1                                   3  13-MAR
[KEMDIR.HALMA.COM]ASSIGNMENT.INC;4                        1  19-AUG
[KEMDIR.HALMA.COM]AUTOINSTALL.COM;14                      7  19-AUG
[KEMDIR.HALMA.COM]COM_TO_HLM.COM;14                       5  23-APR
[KEMDIR.HALMA.COM]EDI.COM;21                              2  24-APR
[KEMDIR.HALMA.COM]EDTINI.EDT;2                            2  24-APR
[KEMDIR.HALMA.COM]HALMA.COM;122                          39   9-SEP
[KEMDIR.HALMA.COM]HALMABATCH.COM;26                      19  24-JUN
[KEMDIR.HALMA.COM]HALMALOGIN.COM;1                        6  18-AUG
[KEMDIR.HALMA.COM]HDLEDTINI.EDT;14                        4  29-APR
[KEMDIR.HALMA.COM]HILONET.COM;1                           3   8-AUG
[KEMDIR.HALMA.COM]HLMEDTINI.EDT;3                         4  24-APR
[KEMDIR.HALMA.COM]HLMLOGIN.COM;45                         6  19-AUG
[KEMDIR.HALMA.COM]HPPLOT.COM;6                            6  13-MAY
[KEMDIR.HALMA.COM]INSTALL.COM;3                           4  27-MAY
[KEMDIR.HALMA.COM]SIMONET.COM;2                           3   9-SEP
[KEMDIR.HALMA.COM]TEKINIT.COM;30                          7  18-AUG
[KEMDIR.HALMA.COM]TEKPLOT.COM;7                           6  27-MAY
[KEMDIR.HALMA]DAT.DIR;1                                   1  17-MAR
[KEMDIR.HALMA.DAT]GDVM.DAT;8                              5  19-MAR
[KEMDIR.HALMA.DAT]GDVMGATES.DAT;1                         5  19-MAR
[KEMDIR.HALMA.DAT]HALMADF.DAT;11                          1  14-JAN
[KEMDIR.HALMA.DAT]SETUP.TREE;1                            1  24-JUN
[KEMDIR.HALMA]DOC.DIR;1                                   1  13-MAR
[KEMDIR.HALMA.DOC]BUGS.DOC;2                              3  13-AUG
[KEMDIR.HALMA.DOC]BUGS.DOC;1                              3  28-MAY
[KEMDIR.HALMA.DOC]HPPLOT.DOC;2                            3   6-MAY
[KEMDIR.HALMA.DOC]MANUAL.TEX;7                          119  13-MAY
[KEMDIR.HALMA.DOC]TEKPLOT.DOC;1                           3  26-NOV
[KEMDIR.HALMA]EXE.DIR;1                                   1  13-MAR
[KEMDIR.HALMA.EXE]HALMA.EXE;1                           322  14-AUG
[KEMDIR.HALMA.EXE]HALMADRUCK.EXE;73                     256  26-MAY
[KEMDIR.HALMA.EXE]HALMAPLOT.EXE;77                      256  26-MAY
[KEMDIR.HALMA.EXE]HALMAPLOTNEU.EXE;21                   256  26-MAY
[KEMDIR.HALMA.EXE]HDLRUN.EXE;8                          208  27-MAR
[KEMDIR.HALMA.EXE]HILONET.EXE;1                         107   8-AUG
[KEMDIR.HALMA.EXE]HPPLOT.EXE;18                          15   6-MAY
[KEMDIR.HALMA.EXE]SIMONET.EXE;4                         102   9-SEP
[KEMDIR.HALMA.EXE]TEKPLOT.EXE;1                          29  13-MAY
[KEMDIR.HALMA.EXE]TREE.EXE;1                            835  11-AUG
[KEMDIR.HALMA]USR.DIR;1                                   3  24-MAR
[KEMDIR.HALMA.USR]BSPBARN.HLM;3                           1  25-APR
[KEMDIR.HALMA.USR]BSPBRA1.HLM;3                           3  27-APR
[KEMDIR.HALMA.USR]BSPCOMPA.HLM;1                          2  23-APR
[KEMDIR.HALMA.USR]BSPDA0190.HLM;2                         1  26-APR
[KEMDIR.HALMA.USR]BSPDA0191.HLM;1                         1  23-APR
[KEMDIR.HALMA.USR]BSPEXOR3.HLM;2                          1  23-APR
[KEMDIR.HALMA.USR]BSPEXOR4.HLM;6                          1   7-MAY
```

```
[KEMDIR.HALMA.USR]BSPFADD.HLM;1          2   23-APR
[KEMDIR.HALMA.USR]BSPFADD1.HLM;1         2   23-APR
[KEMDIR.HALMA.USR]BSPMICRO.HLM;1        27   23-APR
[KEMDIR.HALMA.USR]BSPMUX8.HLM;1          1   23-APR
[KEMDIR.HALMA.USR]GDVM.DAT;8             5   19-MAR
[KEMDIR.HALMA.USR]HALMA.HDL;9            1   13-MAY
[KEMDIR.HALMA.USR]HALMA.HLM;5            2   13-MAY
[KEMDIR.HALMA.USR]HALMA.HNET;2           1   13-MAY
[KEMDIR.HALMA.USR]HALMA.PRT;2            6   13-MAY
[KEMDIR.HALMA.USR]HALMA.TNET;2           3   13-MAY
[KEMDIR.HALMA.USR]HALMA_1.HILO;1         6   30-JUN
[KEMDIR.HALMA.USR]HALMA_1.PLT;1         30   13-MAY
[KEMDIR.HALMA.USR]HALMA_1.PRN;2          6   13-MAY
[KEMDIR.HALMA.USR]HALMA_2.HILO;1         3   30-JUN
[KEMDIR.HALMA.USR]HALMA_2.PLT;1         14   13-MAY
[KEMDIR.HALMA.USR]HALMA_2.PRN;2          5   13-MAY
[KEMDIR.HALMA.USR]HPPLOT.DEF;2           1   13-MAY
[KEMDIR.HALMA.USR]NETPARA.DAT;1          1   30-JUN
[KEMDIR.HALMA.USR]STAT.DAT;4             2   13-MAY
[KEMDIR.HALMA.USR]TEKPLOT.DEF;7          1   13-MAY
[KEMDIR.HALMA.USR]TIMER.DAT;2            2   13-MAY

Total of 69 files, 2793 blocks
End of save set
```

```
[KEMDIR.HALMA.USR]BSPFADD.HLM;1              2  23-APR
[KEMDIR.HALMA.USR]BSPFADD1.HLM;1             2  23-APR
[KEMDIR.HALMA.USR]BSPMICRO.HLM;1            27  23-APR
[KEMDIR.HALMA.USR]BSPMUX8.HLM;1              1  23-APR
[KEMDIR.HALMA.USR]GDVM.DAT;8                 5  19-MAR
[KEMDIR.HALMA.USR]HALMA.HDL;9                1  13-MAY
[KEMDIR.HALMA.USR]HALMA.HLM;5                2  13-MAY
[KEMDIR.HALMA.USR]HALMA.HNET;2               1  13-MAY
[KEMDIR.HALMA.USR]HALMA.PRT;2                6  13-MAY
[KEMDIR.HALMA.USR]HALMA.TNET;2               3  13-MAY
[KEMDIR.HALMA.USR]HALMA_1.HILO;1             6  30-JUN
[KEMDIR.HALMA.USR]HALMA_1.PLT;1             30  13-MAY
[KEMDIR.HALMA.USR]HALMA_1.PRN;2              6  13-MAY
[KEMDIR.HALMA.USR]HALMA_2.HILO;1             3  30-JUN
[KEMDIR.HALMA.USR]HALMA_2.PLT;1             14  13-MAY
[KEMDIR.HALMA.USR]HALMA_2.PRN;2              5  13-MAY
[KEMDIR.HALMA.USR]HPPLOT.DEF;2               1  13-MAY
[KEMDIR.HALMA.USR]NETPARA.DAT;1              1  30-JUN
[KEMDIR.HALMA.USR]STAT.DAT;4                 2  13-MAY
[KEMDIR.HALMA.USR]TEKPLOT.DEF;7              1  13-MAY
[KEMDIR.HALMA.USR]TIMER.DAT;2                2  13-MAY

Total of 69 files, 2793 blocks
End of save set
```